IN THE UNITED STATES PATENT AND TRADEMARK OFFICE

Applicant:	Charles P. Roth et al.	Art Unit:	2183
Serial No.:	09/704,467	Examiner:	Aimee J. Li
Filed:	October 31, 2000	Assignee:	Intel Corporation
Title:	EFFICIENT EMULATION INSTRUCTION DISPATCH BASED ON INSTRUCTION WIDTH		

Mail Stop Appeal Brief - Patents
Commissioner for Patents
P.O. Box 1450
Alexandria, VA 22313-1450

BRIEF ON APPEAL

Applicant hereby files this Brief on Appeal to perfect the Notice of Appeal filed July 11, 2006.

(1) Real Party in Interest

This case is assigned of record to Intel Corporation.

(2) Related Appeals and Interferences

There are no known related appeals and/or interferences.

(3) Status of Claims

Claims 1, 3, 5-9, 11-16, 18-21, and 23-31 are pending. Claims 1, 9, 16, and 21 are in independent form.

CERTIFICATE OF MAILING BY FIRST CLASS MAIL

I hereby certify under 37 CFR §1.8(a) that this correspondence is being deposited with the United States Postal Service as first class mail with sufficient postage on the date indicated below and is addressed to the Commissioner for Patents, P.O. Box 1450, Alexandria, VA 22313-1450.

November 13, 2006
Date of Deposit

Julie H. Giordano
Signature

Julie H. Giordano
Typed or Printed Name of Person Signing Certificate

(4) Status of Amendments

All claim amendments have been entered.

(5) Summary of Claimed Subject Matter

Background: An emulation system can be connected to a data processor to perform procedures such as debugging, hardware development, or software development. *See, e.g., specification*, page 9, line 4-5. Referring to FIG. 5, in order to perform such emulation, instructions can be scanned into a emulation instruction register (such as EMUIR 505) using a test interface (such as a JTAG interface 504). *See, e.g., specification*, page 9, line 7-9. After the instructions are loaded into the instruction register, the test interface may enter a run-test idle (RTI) state to trigger the entry of the instructions into a connected data processor. *See, e.g., specification*, page 10, line 6-9.

Claim 9: Claim 9 relates to a method of providing instructions to a processor. The method includes loading a plurality of instructions into an emulation instruction register from a test interface (*see, e.g.*, page 9, line 7-9; page 9, line 11 - page 10, line 5; page 11, line 19 - page 12, line 4), receiving a run-test idle state signal (*see, e.g.*, page 10, line 6-9; FIG. 6, element 610; page 13, line 1-2), providing the plurality of instructions to the processor in response to the receipt of the run-test idle state signal (*see, e.g.*, page 10,

line 9-19; FIG. 6, element 615), and processing the plurality of instructions without receiving another run-test idle state signal. *See, e.g.*, page 10, line 20-page 11, line 2; page 14, line 4-7. The run-test idle state signal indicates entry of the test interface into a run-test idle state. *See, e.g.*, page 10, line 6-9; FIG. 6, element 610; page 13, line 1-2.

Claim 16: Claim 16 relates to a processor that includes a test interface (*see, e.g.*, FIG. 5, joint test action group (JTAG) interface 504; page 8, line 13-16), an emulation instruction register adapted to store a plurality of emulation instructions received from the test interface (*see, e.g.*, FIG. 5, emulation instruction register (EMUIR) 505; page 9, line 7-9)), emulation control logic adapted to supply the plurality of emulation instructions to a processor pipeline in response to detection of an entry of the test interface into run-test idle state, (*see, e.g.*, FIG. 5, emulation control logic 522; page 10, line 9-19; page 11, line 6-10; page 12, line 7-12) and a decoder to receive the plurality of instructions for processing. *See, e.g.*, FIG. 5, decoder 530; page 12, line 13-16.

Claim 21: Claim 21 relates to an apparatus that includes operating instructions residing on a machine-readable storage medium. The operating instructions are for use in a device to handle a plurality of emulation instructions. The operating instructions cause the device to load the plurality of emulation

instructions into a single emulation instruction register (*see, e.g.*, page 9, line 11 - page 10, line 5), have a test interface enter a run-test idle state (*see, e.g.*, page 10, line 6-9), provide the plurality of emulation instructions to a processor in response to entry of the test interface into the run-test idle state (*see, e.g.*, page 10, line 9-19), and process the plurality of emulation instructions. *See, e.g.*, page 11, line 1-2.

Claim 1: Claim 1 relates to a method that includes receiving a plurality of instructions from a test interface (*see, e.g.*, page 9, line 7-9; page 9, line 11 - page 10, line 5), loading the plurality of instructions into an emulation instruction register (*see, e.g.*, page 9, line 7-9; page 9, line 11 - page 10, line 5; page 11, line 19 - page 12, line 4), receiving a plurality of instructions from the emulation instruction register (*see, e.g.*, page 10, line 9-10; page 12, line 5-10), determining a validity of a first instruction of the plurality of instructions by reading width bits in the first emulation instruction (*see, e.g.*, page 12, line 12-13; page 13, line 9-19), providing the first instruction to a decoder of the processor if the first instruction is valid (*see, e.g.*, page 13, line 20-22; FIG. 6, element 620), without receiving a run-test idle state signal, determining a validity of a second instruction of the plurality of instructions by reading width

bits in the second instruction (*see, e.g.*, page 14, line 4-11), and providing the second instruction to the decoder if the second instruction is valid. *See, e.g.*, page 14, line 11-14; FIG. 6, element 630. Read width bits define the validity and size of the first and second emulation instructions. *See, e.g.*, page 13, line 9-11.

(6) Grounds of Rejection

Claims 9, 16, 21, and 29 stand rejected under 35 U.S.C. § 102(b) as anticipated by U.S. Patent No. 5,530,804 to Edgington et al. (hereinafter "Edgington").

Claims 1, 3, 5-8, 25, and 30-31 stand rejected under 35 U.S.C. § 103(a) as obvious over Edgington and U.S. Patent No. 5,774,737 to Nakano et al. (hereinafter "Nakano").

Claims 11-15, 18-20, 23, 26, and 28 stand rejected under 35 U.S.C. § 103(a) as obvious over Edgington and Nakano.

Claims 24 and 27 stand rejected under 35 U.S.C. § 103(a) as obvious over Edgington, Nakano, and U.S. Patent No. 5,970,241 to Deao et al. (hereinafter "Deao").

The following grounds for rejection are presented for appeal.

I. The rejection of independent claim 9 under 35 U.S.C. § 102(b) as anticipated by Edgington.

II. The rejection of independent claim 16 under 35 U.S.C. § 102(b) as anticipated by Edgington.

III. The rejection of independent claim 21 under 35 U.S.C. § 102(b) as anticipated by Edgington.

IV. The rejection of independent claim 1 under 35 U.S.C. § 103(a) as obvious over Nakano and Edgington.

(7) Argument

I. THE REJECTION OF INDEPENDENT CLAIM 9 UNDER 35 U.S.C. § 102(B) AS ANTICIPATED BY EDGINGTON IS IMPROPER AND SHOULD BE WITHDRAWN

Claim 9 relates to a method of providing instructions to a processor. The method includes loading a plurality of instructions into an emulation instruction register from a test interface, receiving a run-test idle state signal, providing the plurality of instructions to the processor in response to the receipt of the run-test idle state signal, and processing the plurality of instructions without receiving another run-test idle state signal. The run-test idle state signal indicates entry of the test interface into a run-test idle state.

The rejection contends that Edgington describes the receipt of a run-test idle state signal, the provision of a plurality of instructions to the processor in response to the receipt of the run-test idle state signal, and the processing of the plurality of instructions without receipt of another run-test idle state signal, as recited in claim 9. Applicant respectfully disagrees.

In this regard, Edgington describes a processor 10 that includes a single interface over which both test and operational instructions are loaded into processor 10. *See, e.g., Edgington*, col. 2, line 56-61. Edgington's test controller operates in conjunction with this "normal" bus interface to exchange data, address, and control information with processor 10 as in normal (non-test) operations so as to not increase the footprint and cost of test equipment. *See, e.g., Edgington*, col. 4, line 53-59; col. 1, line 17-27.

Perhaps because Edgington is primarily concerned with transitioning this single interface between test instructions and operational instructions, Edgington does not provide many details regarding the provision or processing of test instructions in a test state. Indeed, it appears that only descriptions of an emulation operation provided by Edgington are as follows:

> "An in-circuit emulator/debugger/tester recognizes that the processor 10 has entered the halted state and takes control, in this example, to observe and control processor operation in an emulator mode (although various testing and debugging can also be performed). Step 86, described below, details an example of an in-circuit emulator operation, utilizing the serial shift capability to input commands and data into the processor. Testing and debugging in this mode does not have to be serially shifted into processor 10 via controller 21 (see FIG. 1), but can run instructions from test/debug address space in FIG. 3 using existing hardware of

> processor 10 (existing address bus, data bus, control, pipelines, caches, units, etc.) at the full speed of processor 10." *See Edgington*, col. 14, line 45-59.

As for Edgington's step 86:

> "Step 86 depicts an example of an in-circuit emulator operation in more detail than the 7th step of step 84. In this example, the emulator observes the operational state of the processor (by reading memory locations through usage of the address and data buses of FIG. 1 [in test address space] for viewing). The emulator may also alter configurations within the processor by exercising the override disable capability discussed above, utilizing the serial shift capability of the processor 10. The emulator may modify memory contents by writing memory locations through usage of the address and data buses. At the conclusion of emulator operation, the emulator issues a "restart the CPU" command, serially shifted into the processor 10 as illustrated in FIG. 2, which invokes the processor 10 to resume operation from exactly where the instruction stream was originally halted. *See Edgington*, col. 15, line 13-27.

Nothing in Edgington describes or suggests any relationship between the provision and processing of a plurality of test instructions and the receipt of a run-test idle state signal, as recited in claim 9. Indeed, Edgington appears to lack any signals whatsoever that indicate of the state of the single test/operational interface, much less a run-test idle state signal that indicates entry of the test interface into a run-test idle state.

The rejection contends that Edgington's generate debug mode interrupt (GDMI) signal constitutes such a run-test idle state signal. Applicant respectfully disagrees. Edgington expressly describes that the GDMI signal is "invoked through a GDMI command, execution of a trace or breakpoint, or an external reset." *See Edgington*, col. 3, line 17-20. *See also Edgington*, FIG. 9, elements 72, 74; col. 11, line 20-26; col. 12, line 18-38 (describing that test controller 21 must wait until an interruptible point in the instruction execution stream must be reached before the instruction execution stream can be interrupted and testing can begin). None of these invocative events has anything to do with the state of Edgington's single test/operational interface.

Indeed, test controller 21's receipt of the GDMI signal indicates that testing is to begin. *See, e.g, Edgington*, col. 11, line 5-19 and col. 3, line 17-20. This GDMI signal has nothing to do with Edgington's "normal" bus interface being in a run test idle state. Indeed, to the extent that testing *is only to begin*, the *GDMI signal expressly indicates that the bus interface is not yet in a testing state at all*, but rather in an operational state.

Since Edgington's "normal" bus interface is performing normal operations when the GDMI signal is received, the GDMI signal does not indicate that the "normal" bus interface has entered into a run test idle state. Instead, the GDMI signal *triggers* the interruption of the instruction execution stream.

Therefore, Edgington neither describes nor suggests the receipt of a run-test idle state signal, the provision of a plurality of instructions to the processor in response to the receipt of the run-test idle state signal, and the processing of the plurality of instructions without receipt of another run-test idle state signal, as recited in claim 9. Accordingly, anticipation has not been established. Applicant requests that the rejections of claim 9 and the claims dependent therefrom be withdrawn.

II. THE REJECTION OF INDEPENDENT CLAIM 16 UNDER 35 U.S.C. § 102(B) AS ANTICIPATED BY EDGINGTON IS IMPROPER AND SHOULD BE WITHDRAWN

Claim 16 relates to a processor that includes a test interface, an emulation instruction register adapted to store a plurality of emulation instructions received from the test interface, emulation control logic adapted to supply the plurality of emulation instructions to a processor pipeline in response to detection of an entry of the test interface into run-test idle state, and a decoder to receive the plurality of instructions for processing.

The rejection contends that Edgington describes emulation control logic adapted to supply a plurality of emulation instructions to a processor pipeline in response to detection of an entry of the test interface into run-test idle state. Applicant respectfully disagrees.

In this regard, as discussed above, Edgington's primary concern is the transitioning of a single interface between test instructions and operational instructions. Edgington does not describe or suggest any relationship between the provision of a plurality of test instructions and the entry of a test interface into a run-test idle state, as recited in claim 9. Indeed, Edgington appears to be silent as to the details of any communication between his test controller 21 and the bus interface unit 20.

The rejection contends that Edgington's test controller responds to a generate debug mode interrupt (GDMI) signal by supplying a plurality of emulation instructions to a processor pipeline. Be that as it may, Edgington's GDMI signal does not allow Edgington's test controller to detect an entry of the test interface into run-test idle state. As discussed above, Edgington's GDMI signal is invoked by a GDMI command, execution of a trace or breakpoint, or an external reset. However, none of these events indicate entry of the test interface into a run

test idle state. Indeed, to the extent that Edgington's GDMI signal indicates that testing is to begin, the GDMI signal expressly requires that the bus interface not yet be in a testing state, but rather in an operational state.

Therefore, Edgington neither describes nor suggests emulation control logic adapted to supply a plurality of emulation instructions to a processor pipeline in response to detection of an entry of the test interface into run-test idle state, as recited in claim 16. Accordingly, anticipation has not been established. Applicant requests that the rejections of claim 16 and the claims dependent therefrom be withdrawn.

III. THE REJECTION OF INDEPENDENT CLAIM 21 UNDER 35 U.S.C. § 102(B) AS ANTICIPATED BY EDGINGTON IS IMPROPER AND SHOULD BE WITHDRAWN

Claim 21 relates to an apparatus that includes operating instructions residing on a machine-readable storage medium. The operating instructions are for use in a device to handle a plurality of emulation instructions. The operating instructions cause the device to load the plurality of emulation instructions into a single emulation instruction register, have a test interface enter a run-test idle state, provide the plurality of emulation instructions to a processor in response to entry of the test interface into the run-test idle state, and process the plurality of emulation instructions.

The rejection contends that Edgington describes operating instructions that cause a device to provide a plurality of emulation instructions to a processor in response to entry of a test interface into the run-test idle state. Applicant respectfully disagrees.

In this regard, as discussed above, Edgington's primary concern is the transitioning of a single interface between test instructions and operational instructions. Edgington does not describe or suggest any relationship between the provision of a plurality of test instructions and the entry of a test interface into a run-test idle state, as recited in claim 21. Indeed, Edgington appears to be silent as to the details of any communication between his test controller 21 and the bus interface unit 20.

The rejection contends that Edgington's test controller responds to a generate debug mode interrupt (GDMI) signal by supplying a plurality of emulation instructions to a processor pipeline. Be that as it may, Edgington's GDMI signal does not relate to the entry of a test interface into a run-test idle state. As discussed above, Edgington's GDMI signal is invoked by a GDMI command, execution of a trace or breakpoint, or an external reset. However, none of these events indicate entry of the test interface into a run test idle state. Indeed, to the

extent that Edgington's GDMI signal indicates that testing is to begin, the GDMI signal expressly requires that the bus interface not yet be in a testing state, but rather in an operational state.

Therefore, Edgington neither describes nor suggests operating instructions that cause a device to provide a plurality of emulation instructions to a processor in response to entry of a test interface into the run-test idle state, as recited in claim 21. Accordingly, anticipation has not been established. Applicant requests that the rejections of claim 21 and the claims dependent therefrom be withdrawn.

IV. THE REJECTION OF INDEPENDENT CLAIM UNDER 35 U.S.C. § 103(A) AS OBVIOUS OVER NAKANO AND EDGINGTON IS IMPROPER AND SHOULD BE WITHDRAWN

Claim 1 relates to a method that includes receiving a plurality of instructions from a test interface, loading the plurality of instructions into an emulation instruction register, receiving a plurality of instructions from the emulation instruction register, determining a validity of a first instruction of the plurality of instructions by reading width bits in the first emulation instruction, providing the first instruction to a decoder of the processor if the first instruction is valid, without receiving a run-test idle state signal, determining a validity of a second instruction of the plurality of instructions by reading width bits in the second

instruction, and providing the second instruction to the decoder if the second instruction is valid. Read width bits define the validity and size of the first and second emulation instructions.

The rejection of claim 1 contends that Nakano describes determining a validity of an instruction by reading width bits in the instruction, wherein the width bits define the validity and size of the instruction. In particular, the rejection is understood to contend that certain of Nakano's VLIW instructions (i.e., a VLIW-instruction word length rewrite instruction) are such width bits.

Applicant respectfully disagrees. As discussed in Nakano, a VLIW-instruction word length rewrite instruction causes the instruction word length stored in a word length register 11 to be rewritten. *See, e.g., Nakano*, col. 2, line 25-45. This rewritten instruction word length allows subsequent VLIW-instruction words to be handled. *See, e.g., Nakano*, col. 9, line 18-25.

Nakano's instruction word length rewrite instructions thus do not define the validity and the size of the instruction in which they are found, as required of the width bits in claim 1. Instead, Nakano's instruction word length rewrite instructions allow subsequent instructions to be handled using the rewritten word length.

Edgington does nothing to remedy this deficiency of Nakano. In particular, Edgington neither describes nor suggests such width bits, or even that the size of emulation instructions can change.

In the Office action mailed April 11, 2006, the Response to Arguments contended that Edgington involved VLIW instructions of different sizes. *See Office action mailed April 11, 2006*, page 13, para. 29. However, the citations therein are clearly directed to Nakano rather than Edgington, and clearly do not support the Examiner's contention that Edgington somehow involves instructions of differing sizes, much less instructions that include width bits.

The rejection of claim 1 also contends that Edgington describes determining a validity of a second instruction without receiving a run-test idle state signal, as recited in claim 1. Applicant respectfully disagrees.

As discussed above, Edgington's primary concern is the transitioning of a single interface between test instructions and operational instructions. Edgington does not describe or suggest any relationship between the determination of a validity of an instruction and the receipt a run-test idle state signal, as recited in claim 1. Indeed, Edgington appears to be silent as to the details of any communication between his test controller 21 and the bus interface unit 20.

The rejection contends that Edgington's generate debug mode interrupt (GDMI) somehow constitutes such a run-test idle state signal. Applicant respectfully disagrees. As discussed above, Edgington's GDMI signal is invoked by a GDMI command, execution of a trace or breakpoint, or an external reset. However, none of these events indicate entry into a run test idle state. Indeed, to the extent that Edgington's GDMI signal indicates that testing is to begin, the GDMI signal expressly requires that the bus interface not yet be in a testing state, but rather in an operational state.

Nakano likewise has nothing to do with a run-test idle state signal. Indeed, none of the rejections has ever contended otherwise.

Since elements and/or limitations of claim 1 are neither described nor suggested by Nakano and Edgington, a *prima facie* case of obviousness has not been established. Applicant therefore requests that the rejections of claim 1 and the claims dependent therefrom be withdrawn.

Please apply the brief fee of $500 and any other charges or credits to Deposit Account No. 06-1050.

Respectfully submitted,

Date: November 13, 2006

Scott C. Harris
Reg. No. 32,030

BY
JOHN F. CONROY
REG. NO. 45,485

Fish & Richardson P.C.
12390 El Camino Real
San Diego, California 92130
Telephone: (858) 678-5070
Facsimile: (858) 678-5099

SCH/JFC/jhg
10676423.doc

Appendix of Claims

1. A method comprising:

receiving a plurality of instructions from a test interface;

loading the plurality of instructions into an emulation instruction register;

receiving a plurality of instructions from the emulation instruction register;

determining a validity of a first instruction of the plurality of instructions by reading width bits in the first emulation instruction, the width bits which are read defining the validity and size of the first emulation instruction;

providing the first instruction to a decoder of the processor if the first instruction is valid;

without receiving a run-test idle state signal, determining a validity of a second instruction of the plurality of instructions by reading width bits in the second instruction, the width bits which are read defining the validity and size of the second emulation instruction; and

providing the second instruction to the decoder if the second instruction is valid.

2. (Canceled)

3. The method of Claim 1, further comprising storing the plurality of instructions in the emulation instruction register in subsequent clock cycles.

4. (Canceled)

5. The method of Claim 1, further comprising loading the plurality of instructions in parallel into the emulation instruction register.

6. The method of Claim 1, further comprising providing the second instruction to the decoder after the first instruction is completed.

7. The method of Claim 1, further comprising providing the plurality of instructions to the decoder after a first run-test idle state without entering into a second run-test idle state.

8. The method of Claim 1, further comprising providing the first and second instructions to a digital signal processor.

9. A method of providing instructions to a processor, the method comprising:

loading a plurality of instructions into an emulation instruction register from a test interface;

receiving a run-test idle state signal, the run-test idle state signal indicating entry of the test interface into a run-test idle state;

providing the plurality of instructions to the processor in response to the receipt of the run-test idle state signal; and

processing the plurality of instructions without receiving another run-test idle state signal.

10. (Canceled)

11. The method of Claim 9, further comprising determining a validity of each of the plurality of instructions before processing by reading bits in each instruction indicating a width of the instruction.

12. The method of Claim 11, further comprising aborting processing of any invalid instructions and loading a next instruction into the emulation instruction register.

13. The method of Claim 9, further comprising loading a next instruction into the emulation instruction register if a no-operation instruction is loaded.

14. The method of Claim 9, further comprising providing the plurality of instructions to the processor a plurality of times without reloading the instruction register.

15. The method of Claim 9, further comprising providing the plurality of instructions to a digital signal processor.

16. A processor comprising:

a test interface;

an emulation instruction register adapted to store a plurality of emulation instructions received from the test interface;

emulation control logic adapted to supply the plurality of emulation instructions to a processor pipeline in response to detection of an entry of the test interface into a run-test idle state; and

a decoder to receive the plurality of instructions for processing.

17. (Canceled)

18. The processor of Claim 16, wherein the emulation control logic determines a validity of the plurality of instructions by reading bits in each instruction indicating a width of each instruction and discards any invalid instructions.

19. The processor of Claim 16, wherein the emulation control logic loads a next instruction from the emulation instruction register immediately after detecting a no-operation instruction.

20. The processor of Claim 16, wherein the processor is a digital signal processor.

21. An apparatus, including operating instructions residing on a machine-readable storage medium, for use in a device to handle a plurality of emulation instructions, the operating instructions causing the device to:

load the plurality of emulation instructions into a single emulation instruction register;

have a test interface enter a run-test idle state;

provide the plurality of emulation instructions to a processor in response to entry of the test interface into the run-test idle state; and

process the plurality of emulation instructions.

22. (Canceled)

23. The apparatus of Claim 21, wherein a validity of each of the plurality of instructions is determined before processing by reading bits in each instruction indicating a width of each instruction.

24. The method of Claim 1, further comprising:

scanning instructions from an in-circuit emulator (ICE) to the test interface, the test interface comprising a Joint Test Action Group (JTAG) interface.

25. The method of Claim 1, wherein a pre-determined set of width bits indicates an instruction is invalid.

26. The processor of Claim 16, wherein the emulation instruction register comprises first and second registers.

27. The processor of Claim 16, wherein the emulation control logic comprises a state machine.

28. The processor of Claim 16, further comprises a multiplexer to select between an instruction for the plurality of instructions to send to the processor pipeline.

29. The apparatus of Claim 21, further comprising an in-circuit emulator to monitor operations of the processor.

30. The method of Claim 1, further comprising executing at least one of the plurality of instructions to monitor operation of the processor.

31. The method of claim 1, further comprising:

performing a debugging operation using the first and second instructions.

Evidence Appendix

None

Related Proceedings Appendix

None